НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

із лабораторної роботи №*5*

із дисципліни «Основи програмування»

на тему

«Типи даних, визначених користувачем»

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студент групи КМ-73 | *асистент Громова В.В.* |
| *Садченко М.В.* |  |

Київ — 2018

ЗМІСТ

[ВСТУП 3](#_Toc510489750)

[1 ПОСТАНОВКА ЗАДАЧІ 3](#_Toc510489751)

[1.1 Мета роботи 3](#_Toc510489752)

[1.2 Варіант та завдання лабораторної роботи 3](#_Toc510489753)

[2 ОПИС ПРОГРАМИ 3](#_Toc510489754)

[2.1 Опис коду 3](#_Toc510489755)

[2.3 Опис бібліотек 4](#_Toc510489756)

[ВИСНОВКИ 5](#_Toc510489757)

[ПЕРЕЛІК ПОСИЛАНЬ 5](#_Toc510489758)

[ДОДАТОК А (Відповіді на контрольні запитання) 5](#_Toc510489759)

[ДОДАТОК Б (Код програми) 8](#_Toc510489760)

[ДОДАТОК В (Результат роботи програми) 10](#_Toc510489761)

[ДОДАТОК Г (Блок-схема) 11](#_Toc510489762)

# ВСТУП

Даний звіт створено для детального ознайомлення з лабораторною роботою №5, її завданням, структурою, розробкою програмного коду мовою програмування C, у супроводі детального розбору поставлених вимог.

# 1 ПОСТАНОВКА ЗАДАЧІ

## 1.1 Мета роботи

Вивчити синтаксис і правила роботи з типами даних, визначеними користувачем.

## 1.2 Варіант та завдання лабораторної роботи

Завдання з варіанту №24:

1. Описати структуру з іменем FLAT, що містить такі поля:

* прізвище та ініціали власника;
* площа;
* кількість кімнат;
* етаж.

1. Написати програму, що виконує такі дії:

* уведення з клавіатури даних у масив із 5 структур типу FLAT;
* упорядкування записів за алфавітом прізвищ власників;
* виведення даних щодо квартир з площею менше 30 кв.м і підрахувати їх кількість; якщо таких квартир немає, видати на екран відповідне повідомлення.

# 2 ОПИС ПРОГРАМИ

## 2.1 Опис коду

Програма починається з того, що користувачеві надається вибір між доступом до самовільного заповнення структури (кнопка «1»), очисткою консолі («8») та виходом з програми («9»). Реалізовано за допомогою циклу do-while, функцій strchr (перевіряє, чи входить введений символ в заданий масив) та getch (<conio.h>) (за її допомогою компьютер отримує символ одразу після натискання; користувачеві непотрібно підтверджувати свій вибір Enter`ом). Далі правильний символ надходить до switch-case, що розмежує кожну з дій окремою підпрограмою. Для того, щоб з одного ‘кейсу’ програма просто не переходила у інший – в кінці кожної з підпрограм ставиться break.

Очистка консолі виконується за допомогою команди system(“cls”) з бібліотеки <stdlib.h>.

Вихід з програми реалізовано командою exit(0), теж з <stdlib.h>.

Програма починається з того, що користувач заповнює 5 копій структур, що складається з таких змінних як: int area, rooms, floor, char \*name[60]. Всі дані перевіряються. Для ‘валідації’ натуральних чисел (функція int valid\_int()) використовуються, окрім циклу while, ще такі функції, як strlen() (рахує кількість елементів у масиві), isdigit() (перевіряє числове значення(чи є елемент цифрою)) та atoi() (конвертує строковий тип до цілих чисел). Для перевірки буквених значень (функція char \*valid\_str()) використовується та сама схема, за виключенням того, що замість isdigit() 🡪 isalpha().

Коли всі числа введені, структури наповнено, то комп’ютер у функції ‘permutation()’ починає переставляти місцями, якщо потрібно, для сортування фамілій по алфавіту. Також ще підраховується число квартир, які мають площу, меншу за 30 кв.метрів. Далі це число квартир разом з відсортованим масивов виводиться користувачеві на екран.

Після закінчення кожної програми (окрім виходу з програми) відбувається повтор, що дозволяє ще раз обрати між заданими діями, натиснувши «1», «8» чи «9». У всіх функціях повторювані дії здійснювалися за допомогою нескінченного циклу while(1).

## 2.3 Опис бібліотек

Для початку, опишемо використані бібліотеки у програмі:

<stdio.h> - відповідає за ввод\вивід даних;

<stdlib.h> - відповідає за функції, що займаються виділенням пам'яті, контроль процесу виконання програми, перетворення типів та інші;

<string.h> - містить функції для роботи з нуль-термінованими рядками і різними функціями роботи з пам'яттю;

<ctype.h> - містить оголошення функцій для класифікації символів;

<conio.h> - застосовується для створення текстового інтерфейсу;

<locale.h> – файл оголошує структуру та її поля, які збурігають інформацію про поточне кодування для форматування чисел.

# ВИСНОВКИ

Під час виконання лабораторної роботи були вивчені нюанси роботи із типами даних у мові програмування С. Було застосовано структури, що дозволило нам краще засвоїти цю тему.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Програмування мовою С. Методичні вказівки до виконання лабораторних робіт із кредитного модуля «Програмування-2. С.» для студентів спеціальності 113 «Прикладна математика» [Текст] / В. В. Громова. — К. : НТУУ «КПІ», 2016. —5- 94 с.

# ДОДАТОК А (Відповіді на контрольні запитання)

1. *Для чого потрібні типи, визначені користувачем?*

Для багаторазового застосування декілької змінних.

1. *Які типи відносять до простих типів, а які до складених?*

Програміст під час написання програми мовою С може створювати дані п’ятьох типів:

* структуру (structure);
* об’єднання (union);
* перераховуваний тип (enumeration);
* поля бітів (bit fields);
* нове ім’я (псевдонім) для вже існуючого типу (за допомогою оператора typedef).

1. *Як оголошують структури, чим вони відрізняються від масивів?*

struct student {

char name[30];

char group[3];

int scholarship;

};

struct student stud1year[200];

Масиви містять у собі якийсь один вид змінних та не можуть дублюватися, навідміну від структур.

1. *Способи опису структури.*

У розробленні програм важливим є вибір ефективного способу подання інформації. У багатьох випадках недостатньо оголосити просту змінну чи масив: потрібна гнучкіша форма подання. Таким елементом може бути *структура*, яка дає змогу включати в себе дані різних типів, а також інші структури. Структуру задають у мові С за допомогою ключового слова struct, за яким слідує її ім’я.

1. *Як звертатися до полів структури?*

Один із методів, убудованих у мову C, який дає змогу одержати доступ до біта, — це поля бітів. *Поля бітів* — це спеціальний тип членів структури, у якому визначено, зі скількох бітів складається кожне поле. Основна форма оголошення подібної структури така:

struct *<ім’я\_структури>*

{

*<тип> <ім’я1>* : *<довжина в бітах>*;

*<тип> <ім’я2>* : *<довжина в бітах>*;

*...*

*<тип> <ім’яN>* : *<довжина в бітах>*;

};

де *<тип>* може бути одним із таких: int, unsigned, signed;

будь-яке з *<імен>* може бути пропущено, і тоді відповідна кількість бітів не використується (пропускається).

1. *Які операції допустимі над структурою в цілому?*

Опис, виклик, клонування.

1. *Як здійснюють доступ до елементів структур за допомогою вказівників?*

exam->variable = 5;

1. *Опис вказівника на структуру.*

struct stru\* adr\_pointer;

1. *Як задають масиви структур?*

struct student stud1year[200];

1. *Яким чином структури передають у функції?*

Викликають у тілі функції:

struct student student1;

1. *Чи можна передавати у функції масиви структур?*

Так.

1. *Як оголошують об’єднання, чим воно відрізняється від структури?*

Об’єднання описують так само, як і структури, але замість ключового слова struct використовують ключове слово union. Як і для структур, можна визначити шаблон об’єднання з іменем типу й задати в цьому ж операторі одну чи декілька змінних об’єднання.

1. *Скільки виділяється пам’яті для об’єднання?*

На відміну від структури, для змінної типу union місця в пам’яті виділяється рівно стільки, скільки потрібно полю об’єднання, що має найбільший розмір у байтах.

1. *Які операції допустимі для змінних типу об’єднання?*

Синтаксис використання полів об’єднання такий самий, як і для структури, наприклад, u.ch = 'N'. Для об’єднання також дозволено операцію ->, якщо звернення до об’єднання здійснюють за допомогою вказівника. Для змінних типу об’єднання також можна використовувати операції взяття адреси та одержання елемента через указівник.

1. *Як задають змінні на перерахування?*

*Перерахування* — це конструйований тип даних, під час опису якого задають ім’я типу даних і значення, яких можуть набувати змінні цього типу.

Опис перерахованого типу даних здійснюють оператором виду:

enum *<ім’я\_типу>* {*<список\_значень>*} *<список змінних>*;

де *<ім’я\_типу>* задають ідентифікатором С на розсуд програміста;

*<список\_значень>* задають у вигляді: *<елемент1>*, *<елемент2>*, ..., *<елементN>*, де будь-який із елементів є або *<ім’я>*, або *<ім’я>* = *<константний\_вираз>*. Як *<ім’я>* використовують ідентифікатор мовою С;

*<список\_змінних>* — імена змінних типу перерахування, відокремлені один від одного комами.

1. *Як задають масиви перерахувань?*

enum day week[6];

1. *Для чого застосовують ключове слово typedef?*

Мова С дає змогу дати нову назву вже існуючому типу даних. Для цього використовують ключове слово typedef. При цьому створюється новий тип даних. Наприклад:

typedef char SYMBOL;

typedef unsigned UNSIGN;

typedef float real;

Досить часто використовують оператор typedef із застосуванням структур:

typedef struct st\_tag

{

char name[40];

int year;

char group[5];

int scholarship;

} STUDENT;

Тепер для визначення змінної можна використовувати як struct st\_tag avar, так і STUDENT avar.

# ДОДАТОК Б (Код програми)

#include <stdio.h>

#include <string.h> // strlen(), strchr()

#include <stdlib.h> // atoi(), exit(0), system("cls")

#include <ctype.h> // isdigit()

#include <conio.h> // getch()

#include <locale.h> // для отображения кириллицы

#define STRCT 5

#define WRD\_LEN 60

int valid\_int(int n, int lvl) {

char str[WRD\_LEN];

int num;

if (n == 1) printf("**\n**%d. Area of the flat: ", lvl);

else if (n == 2) printf("%d. Number of rooms: ", lvl);

else if (n == 3) printf("%d. Floor of the flat: ", lvl);

while (1) {

int smb = 0;

scanf("%s", str); //input of value

int len = strlen(str);

for (int i=0; i<len; ++i) {

if (!isdigit(str[i])) {

++smb;

break;

}

}

if (smb == 0) break;

else printf("need natural number!: ");

}

num = atoi(str); //convert str to int

return num;

}

char \*valid\_str(int lvl) {

setlocale(LC\_ALL, "Russian");

static char str[WRD\_LEN];

printf("%d. Enter your last name: ", lvl);

while (1) {

int smb = 0;

scanf("%s", str);

for (int i = 0; i < strlen(str); ++i) {

if (!isalpha(str[i])) {

++smb;

break;

}

}

if (smb == 0) break;

else printf("need literal input!: ");

}

return str;

}

void permutation(char arr[][WRD\_LEN]) {

for (int i = 0; i < STRCT; ++i) {

for (int j = 1; j < STRCT; ++j) {

for (int m = 0; m < STRCT; ++m) {

if (tolower(arr[j-1][m]) > tolower(arr[j][m])) {

for (int k = 0; k < STRCT; ++k) {

char switcher[1][WRD\_LEN];

switcher[0][k] = arr[j-1][k];

arr[j-1][k] = arr[j][k];

arr[j][k] = switcher[0][k];

}

break;

}

else if (tolower(arr[j-1][m]) < tolower(arr[j][m])) break;

}

}

}

printf("**\n**sort array = {");

for (int i = 0; i < STRCT; ++i) {

if (i != STRCT-1) printf("%s, ", arr[i]);

else printf("%s", arr[i]);

}

printf("}**\n**");

}

struct FLAT {

int area, rooms, floor; // этаж

char \*name[WRD\_LEN];

};

int main() {

while (1) {

int choice;

fflush(stdin);

printf("**\n**Press '1' to fill the structure.**\n**"\

"Press '8' to clear console.**\n**"\

"Press '9' to get out program.**\n**");

do {

choice = getch();

} while(!strchr("189", choice));

switch (choice) {

case '1': {

char arr[STRCT][WRD\_LEN];

int num\_area = 0;

struct FLAT flat1 = {valid\_int(1,1), 12, 13, valid\_str(1)};

strcpy(arr[0], \*flat1.name);

struct FLAT flat2 = {valid\_int(1,2), 22, 23, valid\_str(2)};

strcpy(arr[1], \*flat2.name);

struct FLAT flat3 = {valid\_int(1,3), 32, 33, valid\_str(3)};

strcpy(arr[2], \*flat3.name);

struct FLAT flat4 = {valid\_int(1,4), 42, 43, valid\_str(4)};

strcpy(arr[3], \*flat4.name);

struct FLAT flat5 = {valid\_int(1,5), 52, 53, valid\_str(5)};

strcpy(arr[4], \*flat5.name);

permutation(arr);

if (flat1.area < 30) ++num\_area;

if (flat2.area < 30) ++num\_area;

if (flat3.area < 30) ++num\_area;

if (flat4.area < 30) ++num\_area;

if (flat5.area < 30) ++num\_area;

printf("**\n**number of flats with area <30: %d**\n\n**", num\_area);

break;

}

case '8': {

printf(" 7 ballov :)**\n**");

system("cls");

break;

}

case '9': exit(0); break;

default: break;

}

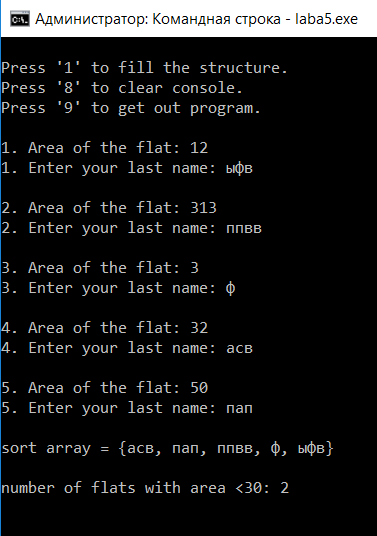
}

return 0;

}

# 

# ДОДАТОК В (Результат роботи програми)



# 

# ДОДАТОК Г (Блок-схема)

